D-036

異種データベースの仮想化技術

Virtualization technology for heterogeneous databases

渡辺 裕太† 菖蒲 佳右† 和田 雄次†† 澤本 潤††† 加藤 貴司†††

Yuta Watanabe Keisuke Syoubu Yuji Wada Jyun Sawamoto Takashi Katoh

1. はじめに

今日では，ユビキタスセンサーネットワーク環境上から大量のデータが収集されており，これらのデータの中に隠された知識や傾向を，データマイニング技術を用いて発見･分析し，業務の意思決定などに役立てることが重要となっている．そして，それらのデータは分散配置された多種多様なデータベース（すなわち，マルチデータベース）に存在する．

しかし，このマルチデータベースに対し，データマイニングを行う分析者は本来的には分析やルール抽出作業に集中したいにも拘わらず，データマイニングの準備過程であるデータベース選択やデータ収集などの作業に多大な時間を費やしてしまうといった課題がある．

そこで，本研究ではこのデータ分析者の負担を軽減するために，ユビキタスコンピューティング環境上にのマルチデータベースが，あたかも一つのデータベースであるかのように利用できるデータベース仮想化技術の開発を目的とする．

データベースにはデータモデルの違いやベンダーの違いによる様々な種類のものが存在する．データモデルの違いでは，データの表現方法が異なり，それらの操作方法もそれぞれ特有である．代表例として表形式のリレーショナルデータベース(RDB)，XML 形式の XML データベース(XMLDB)，オブジェクト指向のオブジェクト指向データベース(OODB)などがある．また，同じモデルのデータベースでも，それぞれのベンダーによる機能の違いがある．例えばRDBにおいてSQLの違いやデータ型の違いなどがある．異種ベンダーRDB の代表的な例として，MySQL, PostgreSQL，SQLServer などが挙げられる．

† 東京電機大学 大学院
Tokyo Denki University Graduate School
†† 東京電機大学 Tokyo Denki University
†† 岩手県立大学 Iwate Prefectural University

これらモデルやベンダーの違いは，いくつかの不都合が生じる．例えば，アプリケーションなどの開発段階において，データモデルの違いにより開発工数が増加してしまう．例えば，複数の異種モデル DB のデータを扱うのであれば，それぞれの API が必要となる．それらの異種モデル DB を仮想化し，手続きを一体化させることによって負荷やコストなどが軽減され，柔軟に管理を行うことができる．以上のことからもデータベースを仮想化することによって，アプリケーションの設計やデータベース管理が容易になると考えられる．

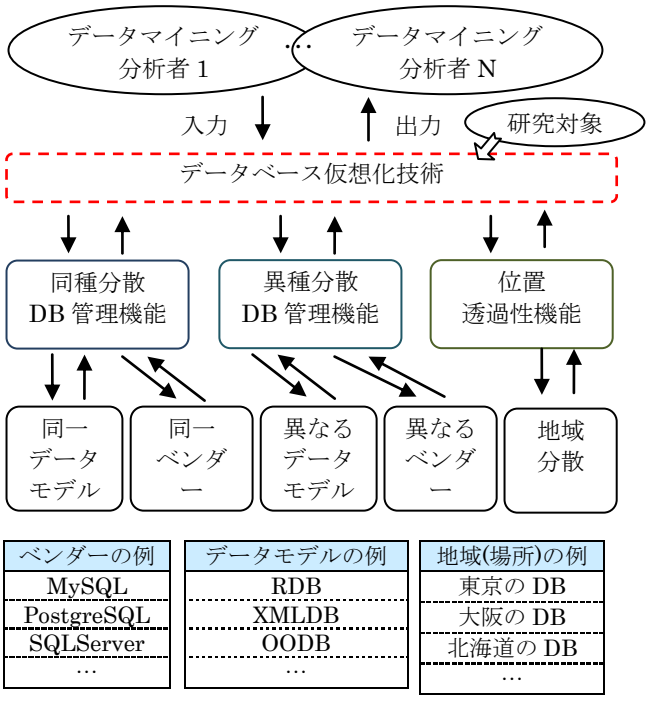

図 1. データベース仮想化技術

仮想化技術として，構造の違いや位置の違いなどを利用者に意識させないため，同種分散データベース管理機能や異種分散データベース管理機能，位置透過性機能といった機能を取り入れ， あらゆるデータベースに対しても柔軟に対応できるようにすることを目的とする．図 1 はそのデータベース仮想化技術のレイヤである．

2. 関連研究

仮想化データベースとして，いくつか行われている関連研究がある．参考文献[1]に記載されている ”モバイルコンピューティング環境におけるユーザ情報の動的軽量における能動型情報配信方式”では，広域ネットワークに接続された異種データベース群を情報源としてモバイルコンピューティング環境上のユーザへ能動的に情報配信するシステムの実現方式を提案している．異種データベースとして，ローカルデータベース群から写像されたデータを，メタデータベースへ構築･検索する基本操作によって異種ローカルデータベース群のデータを統合している．また，参考文献[2]に記載されている RedHat 社が開発した teiid では，同様に異種データベースの仮想化を行うものであり，この仮想化データベースを経由し，リレーショナルデータベース、Web サービス、ERP や CRM などの業務アプリケーションなどさまざまなデータソースにリアルタイムでアクセスし統合できる．teiid は独自の query エンジンを持ちこの query エンジンを中核に、JBDC／SOAP アクセス層を経由してビジネスアプリケーションと，コネクタフレームワークを経由しデータソースと接続することで，リアルタイムのデータ統合が実現されている．

本研究ではメタデータではなく XML スキーマを元にデータベースを仮想化することを提案する．

3. 仮想化技術

本研究では仮想化として，それぞれのスキーマ情報を一つの XML スキーマで表現し，その共通スキーマを元にデータの検索や更新などを行えるようにする．まず，同種分散･異種分散の仮想化を個々に考え，最終的に二つの分散仮想化システムを統合し，あらゆるデータベースに対してもアクセスできるようにする．

3.1. 同種分散

データベース仮想化技術の第一ステップとして，RDB の異なるベンダー間での仮想化データベース管理システムを構築する．仮想化技術として，分析者は意思なく必要なデータだけを期待できるようにする．

3.1.1. XML 変換プログラム

まず，XMLDB やオブジェクト指向 DB といった異種データモデルとの仮想化を行うために，表現能力や移植性のある XML スキーマを利用する．

このために，RDB のスキーマ情報とデータの情報を XML スキーマに変換し，その情報を任意の操作したい RDB に格納することによって，その RDB からあたかも他の RDB を操作しているかのような仮想化の概念を実現させる．今回は，その XMLExport/Import プログラムを作成した．(図 2)

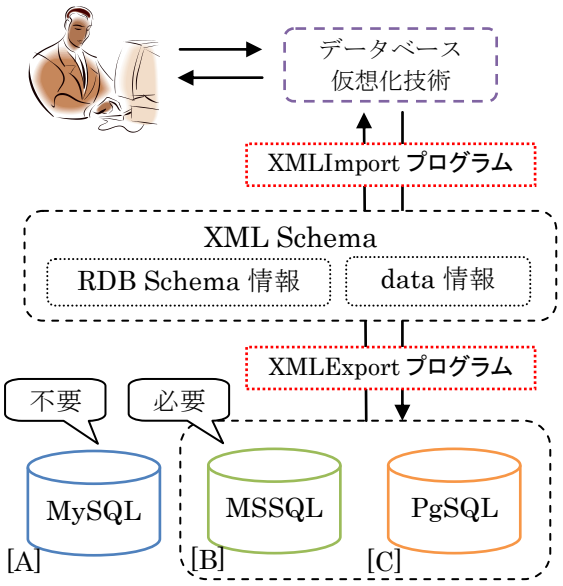

図 2. RDB 仮想化技術

開発環境

今回，RDB 仮想化システムの構築として，異ベンダー RDB をそれぞれ Mysql，PostgreSQL，SQLServer2005 とし，XML Import/Export システムを開発するプログラミング言語として PHP 言語を使用した．

3.1.2. RDB スキーマの XML 変換

図 3 は RDB からスキーマ情報を読み取り，XML に変換したものである．RDB のスキーマ情報として「テーブル名」,「フィールド名(データ型, デフォルト値)」,「制約(主キー制約，ユニーク制約，チェック制約，NOT NULL 制約，外部キー制約)」の情報を XML 形式に変換できるようにした．

XML のツリー構造は，図 4 のようにテーブルの情報は table_structure ノードに記述し，要素としてはそれぞれ Field=”カラム名”，Type=”データ型”，Null=”TRUE or FALSE”(NOT NULL 制約)とした．また，スキーマの情報は schema ノードに記述し，要素としてそれぞれ TYPE=”制約名”，Table=”テーブル名”，Column=”カラム名”，ReTable=”参照元テーブル名”，ReColumn=”参照元カラム名”，Check=”規則”とした．

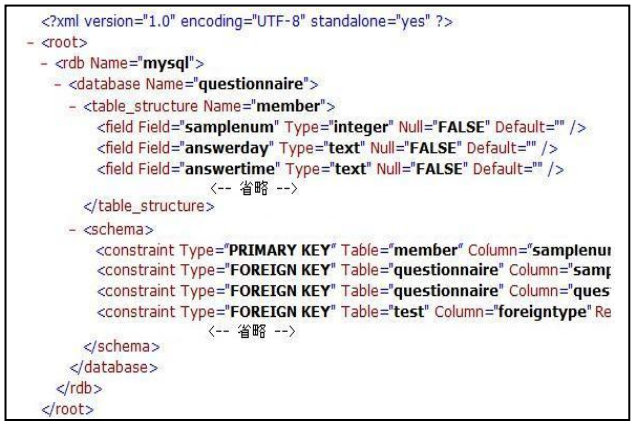

図 3. RDB スキーマ情報の XML 変換例

3.1.3. RDB 実際のデータの XML 変換

図 4 は RDB から実際のデータ情報を読み取り，XML に変換したものである．XML のツリー構造としては dbname=”データベース名”，tblname=”テーブル名”，以降カラム名=”実際のデータ”とした．

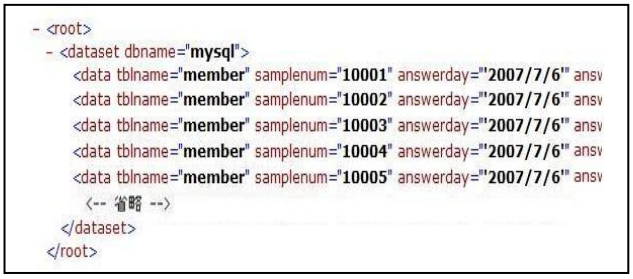

図 4. RDB 実際のデータの XML 変換例

3.2 異種分散

第 2 のステップとして異種モデル DB での仮想化を考える．異種モデル DB の仮想化では，それぞれ違ったモデルのスキーマ情報を 1 つの共通なスキーマで表現する．共通のスキーマとして XML Schema を使用し，それを中枢として仮想化を行う．図 5 に異種データベース仮想化技術のの構造を示す．

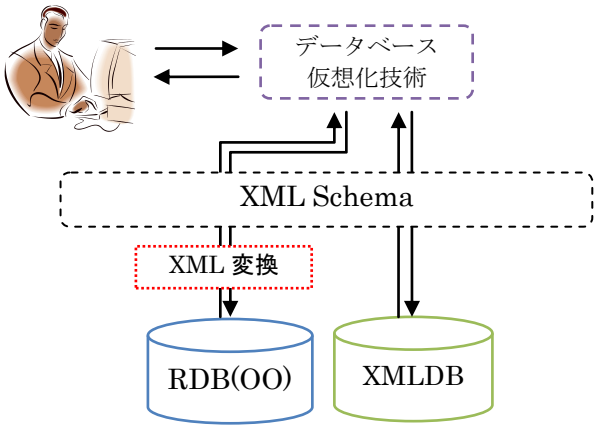

図 5.異種データベース仮想化技術

異モデル DB からのスキーマ変換として，RDB からはスキーマ情報を取得し，それに対応した XML Schema に変換する．表 1 はその対応を表したものである．また，XMLDB では元々XML のデータ形式なので，変換を行わずそのままの形式でスキーマ情報を取得する．OO については，RDB の拡張であるオブジェクト指向の機能を使用する．

表 1　SQL と XML の対応表

	SQL	XML
表定義	CREATE TABLE 表名…	<xsd: element name=”表名”…
列定義	CREATE TABLE… 列名…	<xsd: element name=”列名”…
データ型定義	CREATE TABLE… データ型..	<xsd: element… type=”データ型”…
デフォルト値	CREATE TABLE… 列名 DEFAULT 値	<xsd: element… default=”値”…
主キー制約	PRIMARY KEY	<xsd: key…
一意制約	UNIQUE	<xsd:unique …
外部キー制約	FOREIGN KEY	<xsd: keyref … refer =…
NOT NULL	NOT NUKK	<xsd:… minOccurs=”1”...
CHECK 制約	CHECK(式)	
メソッド	CREATE METHOD	
継承	CREATE TABLE… UNDER 上位表名	<xsd: complexType …

4. 実験と結果

ここでは，同種分散の仮想化を実験した結果を述べる．実際に MySQL と SQLServer の情報データ(スキーマ情報と実際のデータ)を XML スキーマから読み取り Postgre-SQL に格納して，PostgreSQL 側からの操作を試みる．

今回はデータベース名を仮に各 RDB の名前として格納するようにした．図 6 は PostgreSQL 内のデータベース一覧である．「mysql」と「sqlserver」というデータベースが追加されていることがわかる．

```
postgres=# ¥l
        List of databases
    Name    |  Owner   | Encoding
------------+----------+----------
 mysql      | postgres | UTF8
 sqlserver  | postgres | UTF8
 template0  | postgres | UTF8
```

図 6. PostgreSQL 内 DB 一覧

図 7 は PostgreSQL 内のデータベース「mysql」に格納されているテーブル名一覧である．そして，図 8 はその中の「member」テーブルに格納されている実際のデータである．これらは，実際には RDB である MySQL に格納されていたデータベースのテーブルとデータであるが，PostgreSQL 側から操作を行っている．

```
postgres=# ¥connect mysql
You are now connected to database "mysql".
mysql=# SELECT relname FROM pg_stat_user_tables;
    relname
---------------
 choice
 member
 question
 questionnaire
```

図 7. PostgreSQL 内 DB 名 mysql のテーブル名一覧

```
mysql=# SELECT * FROM member;
 samplenum | answerday | answertime
-----------+-----------+---------------
     10001 | 2007/7/6  | 13:07:19:499
     10002 | 2007/7/6  | 13:07:19:499
     10003 | 2007/7/6  | 13:07:19:499
     10004 | 2007/7/6  | 13:07:19:499
     10005 | 2007/7/6  | 13:07:19:499
       …   |     …     |      …
```

図 8. PostgreSQL 内テーブル名 member のデータ

5. 考察と今後の課題

各 RDB からスキーマ情報とデータを取り出し，任意の RDB に格納することによって，その任意の RDB から他の RDB の操作をしているかのような，そういった仮想化の第一歩を実現することができた．

しかし，いくつか相互のデータベース間で対応していないものが存在した．それは，データ型と制約である．データ型に関しては RDB 独自のデータ型がいくつか存在し，それが他とは異なるため，データ型を継承できなかった．今回はそれと近いデータ型を代用した．そして，制約としては MySQL には CHECK 制約が実装されていなかった．対策としては，MySQL で CHECK 制約の処理を行うのではなくプログラム側で CHECK 制約の代わりになるようなものを追加させるということを考えている．

そして 3.2.1 節で述べた 5 つの制約を実装させることができたが，他の制約について，例えばトリガー制約やインデックスなどといったものを取り入れる予定である．

6. おわりに

本研究において，同種分散の仮想化を実装したが，次のステップである異種データモデルの仮想化はまだ開発途中である．この仮想化を実装して 2 つの基盤を統合し，位置透過性の機能を取り入れることによって，あらゆるデータベースにも対応できるようにする．そして最後にそれらについての評価を行う予定である．

7. 参考文献

[1] 森薫 倉林修一 石橋直樹 清木康，“モバイルコンピューティング環境におけるユーザ情報の動的軽量による能動型情報配信方式”

[2] teiid，http://www.jboss.org/teiid，Red Hat

[3] 星野努，「PHP+MySQL最速Webシステム」，技術評論社

[4] 朝羽義之 石田朗雄 稲葉香理 永安悟史，「PostgreSQL徹底入門 第2版」，翔泳社

[5] Microsoft TechNet http://technet.microsoft.com/ja-jp/default.aspx

[6] MySQL，http://www-jp.mysql.com/

[7] PostgreSQL，http://www.postgresql.jp/

[8] SQL Server，http://www.microsoft.com/japan/SQL/default.mspx

[9] PHPプロ！，http://www.phppro

[10] goo Research，http://research.goo.ne.jp/